Amendment to the Claims:

The claims under examination in this application, including their current status and changes made in this paper, are respectfully presented.

1 (canceled).

2 (previously presented). An instruction-programmable processor, comprising:

a program memory for storing instruction opcodes;

a central processing unit, including one or more execution units for executing data processing instructions, and including an instruction fetch unit for presenting a fetch address to the program memory for fetching therefrom an instruction opcode corresponding to the fetch address; and

a loop cache, coupled to the instruction fetch unit, and comprising:

a base address register, for storing a base fetch address;

a branch cache register file, having a plurality of storage locations for storing instruction codes corresponding to a sequence of fetch addresses beginning with the base fetch address, having a data input coupled to the output of the program memory, and having a data output;

a multiplexer, having a first input coupled to an output of the program memory, having a second input coupled to the data output of the branch cache register file, having a select input, and having an output coupled to the instruction fetch unit of the central processing unit; and

loop cache control logic, having a first control output coupled to a control input of the program memory, having a second control output coupled to the select input of the multiplexer, and having a third control output coupled to the branch cache register file for controlling writes thereto and reads therefrom, the loop cache control logic for controlling the multiplexer to select the output of the branch cache register file and for disabling a read of the program memory, responsive to the fetch address corresponding to one of the instruction codes stored in the branch cache register file;

wherein the loop cache control logic also has an input for receiving a backward branch signal indicating that a fetch address corresponds to a backward branch, the loop cache control logic also for controlling the branch cache register file to load an instruction code received at its data input from the program memory responsive to receiving a backward branch signal in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file.

3 (original). The processor of claim 2, further comprising:

backward branch detection logic, comprising a last fetch register for storing a previous fetch address and a comparator for comparing a current fetch address to the contents of the last fetch register, the comparator having an output for generating the backward branch signal responsive to the current fetch address being less than or equal to the previous fetch address.

4 (original). The processor of claim 2, further comprising:

an index comparator, having a first input coupled to the base address register, having a second input coupled to the instruction fetch unit to receive the fetch address therefrom, and having an output coupled to an address input of the branch cache register file and to the loop cache control logic, for presenting an index value corresponding to a difference between the fetch address and the base fetch address.

5 (original). The processor of claim 4, further comprising:

a valid bit register, comprising a plurality of bit locations, each associated with one of the storage locations of the branch cache register file, each for indicating whether the contents of its associated storage location of the branch cache register file contains a valid instruction code.

6 (previously presented). The processor of claim 2, further comprising:

a next candidate register, for storing a fetch address responsive to the loop cache control logic receiving a backward branch signal in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file;

wherein the base address register is coupled to the next candidate register, and is for loading the contents of the next candidate register as a base fetch address responsive to the loop cache control logic receiving a backward branch signal in combination with the fetch address corresponding to the contents of the next candidate register;

and wherein the loop cache control logic is for controlling the branch cache register file to load an instruction code received at its data input from the program memory responsive to receiving a backward branch signal in combination with the fetch address corresponding to the contents of the next candidate register.

7 (previously presented). The processor of claim 6, further comprising:

an index register, having an output coupled to an address input of the branch cache register file, and coupled to an incrementer for incrementing the contents of the index register upon the loop cache receiving each fetch address;

a valid bit register, comprising a plurality of bit locations, each associated with one of the storage locations of the branch cache register file, each for indicating whether the contents of its associated storage location of the branch cache register file contains a valid instruction code;

wherein the loop cache control logic also has an input for receiving a sequential fetch signal indicating that the fetch address from the instruction fetch unit is in sequence with a previous fetch address;

and wherein the loop cache control logic is for controlling the branch cache register file to load an instruction code received at its data input from the program memory, at a storage location indicated by the contents of the index register, responsive to receiving the sequential fetch signal and to the bit location of the valid bit register indicating that the storage location corresponding to the contents of the index register does not contain a valid instruction code.

8 (original). The processor of claim 7, wherein the loop cache control logic is also for controlling the branch cache register file to present, at its output, the contents of a storage location indicated by the contents of the index register, responsive to receiving the sequential fetch signal and to the bit location of the valid bit register indicating that the storage location corresponding to the contents of the index register contains a valid instruction code.

9 (previously presented). The processor of claim 2, wherein the program memory comprises:

a level one instruction memory, having a read control input coupled to the loop cache control logic, and having a data output coupled to the multiplexer;

a level one tag memory, for storing tag addresses corresponding to memory locations for which the level one instruction memory stores instruction codes; and

a level one tag comparator, having an input for receiving the fetch address from the fetch instruction unit, and having an input coupled to the level one tag memory, for comparing the fetch address to the tag addresses to determine whether the fetch address corresponds to a memory location for which the level one instruction memory stores a valid instruction code.

10 (original). The processor of claim 9, wherein the program memory further comprises:

a level two cache, coupled to the level one instruction memory, for storing instruction codes.

11 (previously presented). The processor of claim 10, wherein the program memory and the level two cache are located on the same integrated circuit as the central processing unit.

12 (previously presented). A method of fetching instructions for execution by one or more execution units in a central processing unit of an instruction-programmable processor, comprising:

receiving a sequence of fetch addresses from an instruction fetch unit of the central processing unit;

responsive to one of the received fetch addresses corresponding to a backward branch operation and not corresponding to valid contents of a loop cache, loading a base location of a branch cache register file with an instruction code corresponding to the fetch address from program memory, and setting a valid bit corresponding to the base location;

then, responsive to receiving a fetch address following the backward branch operation and within a storage capacity of the branch cache register file, fetching the instruction code corresponding to the fetch address from program memory, loading the instruction code in a next indexed location of the branch cache register file, and setting a valid bit corresponding to the next indexed location; and

then, responsive to receiving a fetch address corresponding to a location of the branch cache register file for which a corresponding valid bit is set:

disabling the program memory from performing a read access; and

forwarding the contents of the location of the corresponding branch cache register file to the central processing unit.

13 (previously presented). The method of claim 12, further comprising:

after receiving a fetch address, comparing the received fetch address with the contents of a base address register to determine a difference value;

responsive to the difference value being greater than a capacity of the branch cache register file, determining whether the received fetch address corresponds to a backward branch relative to a previous fetch address; and

responsive to the determining step determining that the received fetch address corresponds to a backward branch, loading the received fetch address in the base address register;

wherein the loading step is performed responsive to the determining step determining that the received fetch address corresponds to a backward branch.

14 (original). The method of claim 13, wherein the determining step comprises:

storing a previous fetch address in a previous fetch address register;

comparing the received fetch address to the contents of the previous fetch address register to determine whether the received fetch address is less than the contents of the previous fetch address register.

15 (previously presented). The method of claim 12, further comprising:

after receiving a fetch address, detecting whether the received fetch address corresponds to a backward branch relative to a previous fetch address;

responsive to the detecting step detecting that the received fetch address corresponds to a backward branch, comparing the received fetch address with the contents of a candidate register;

responsive to the comparing step determining that the received fetch address does not match the contents of the candidate register, loading the received fetch address in the candidate register;

wherein the step of loading the base location of a branch cache register file is performed responsive to the comparing step detecting that the received fetch address matches the contents of the candidate register.

16 (previously presented). The method of claim 15, further comprising:

responsive to the comparing step determining that the received fetch address matches the contents of the candidate register, setting an index register;

then, responsive to receiving a next fetch address, repeating the detecting step;

responsive to the detecting step detecting that the received fetch address does not correspond to a backward branch, determining whether the received fetch address corresponds to a sequential fetch; and

responsive to determining that the received fetch address corresponds to a sequential fetch, testing a valid bit corresponding to the contents of the index register;

wherein the step of fetching the instruction code is performed responsive to the resulting of the testing step indicating that the valid bit corresponding to the contents of the index register is not set;

wherein the disabling and forwarding steps are performed responsive to the result of the testing step indicating that the valid bit corresponding to the contents of the index register is set;

and further comprising:

after either the forwarding or fetching steps, incrementing the index register.

17 (previously presented). The method of claim 16, further comprising:

responsive to determining that the received fetch address does not correspond to a sequential fetch, fetching the instruction code corresponding to the fetch address from program memory.